```
RRRRRRRRRRRR     MMM         MMM     SSSSSSSSSSSS
RRRRRRRRRRRR     MMM         MMM     SSSSSSSSSSSS
RRRRRRRRRRRR     MMM         MMM     SSSSSSSSSSSS
RRR         RRR  MMMMMM   MMMMMM  SSS
RRR         RRR  MMMMMM   MMMMMM  SSS
RRR         RRR  MMMMMM   MMMMMM  SSS
RRR         RRR  MMM   MMM   MMM  SSS
RRR         RRR  MMM   MMM   MMM  SSS
RRR         RRR  MMM   MMM   MMM  SSS
RRRRRRRRRRRR     MMM         MMM     SSSSSSSSS
RRRRRRRRRRRR     MMM         MMM     SSSSSSSSS
RRRRRRRRRRRR     MMM         MMM     SSSSSSSSS
RRR   RRR        MMM         MMM              SSS
RRR   RRR        MMM         MMM              SSS
RRR   RRR        MMM         MMM              SSS
RRR      RRR     MMM         MMM              SSS
RRR      RRR     MMM         MMM              SSS
RRR      RRR     MMM         MMM              SSS
RRR         RRR  MMM         MMM  SSSSSSSSSSSS
RRR         RRR  MMM         MMM  SSSSSSSSSSSS
RRR         RRR  MMM         MMM  SSSSSSSSSSSS
```

```
_$2

Sy
---
NT$
NT$
NT$
NT$
NT$
NT$


NT$
NT$
NT$
NT$
NT$
NT$
NT$
NT$
NT$
NT$
NT$
NT$
NT$
NT$
NT$
NT$
NT$
NT$
NT$
NT$
NT$

NT$

NT$
NT$
NT$
NT$
NT$
NT$
NT$



NT$
NT$
NT$
NT$
NT$
NT$
PI
```

```
**FILE**ID**RM3OPEN
```

```
RRRRRRRR    MM      MM    333333      000000    PPPPPPPP    EEEEEEEEEE  NN      NN
RRRRRRRR    MM      MM    333333      000000    PPPPPPPP    EEEEEEEEEE  NN      NN
RR      RR  MMMM  MMMM  33      33  00      00  PP      PP  EE          NN      NN
RR      RR  MMMM  MMMM  33      33  00      00  PP      PP  EE          NN      NN
RR      RR  MM  MM  MM          33  00      00  PP      PP  EE          NNNN    NN
RR      RR  MM  MM  MM          33  00      00  PP      PP  EE          NNNN    NN
RRRRRRRR    MM      MM        33    00      00  PPPPPPPP    EEEEEEEE    NN  NN  NN
RRRRRRRR    MM      MM        33    00      00  PPPPPPPP    EEEEEEEE    NN  NN  NN
RR  RR      MM      MM          33  00      00  PP          EE          NN    NNNN
RR  RR      MM      MM          33  00      00  PP          EE          NN    NNNN
RR    RR    MM      MM  33      33  00      00  PP          EE          NN      NN      ....
RR    RR    MM      MM  33      33  00      00  PP          EE          NN      NN      ....
RR      RR  MM      MM    333333      000000    PP          EEEEEEEEEE  NN      NN      ....
RR      RR  MM      MM    333333      000000    PP          EEEEEEEEEE  NN      NN      ....

LL            IIIIII      SSSSSSSS
LL            IIIIII      SSSSSSSS
LL              II      SS
LL              II      SS
LL              II      SS
LL              II      SS
LL              II        SSSSSS
LL              II        SSSSSS
LL              II              SS
LL              II              SS
LL              II              SS
LL              II              SS
LLLLLLLLLL    IIIIII    SSSSSSSS
LLLLLLLLLL    IIIIII    SSSSSSSS
```

RM3
V04

```
0 MODULE RM3OPEN (LANGUAGE (BLISS32) ,
0                 IDENT = 'V04-000'
0                 ) =
1 BEGIN
1 !
1 !****************************************************************************
1 !*                                                                          *
1 !*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                 *
1 !*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                  *
1 !*  ALL RIGHTS RESERVED.                                                    *
1 !*                                                                          *
1 !*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED   *
1 !*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE   *
1 !*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER   *
1 !*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY   *
1 !*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY   *
1 !*  TRANSFERRED.                                                            *
1 !*                                                                          *
1 !*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE   *
1 !*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT   *
1 !*  CORPORATION.                                                            *
1 !*                                                                          *
1 !*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS   *
1 !*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                 *
1 !*                                                                          *
1 !*                                                                          *
1 !****************************************************************************
```

```
1 !++
1 !
1 ! FACILITY:      RMS32 INDEX SEQUENTIAL FILE ORGANIZATION
1 !
1 ! ABSTRACT:
1 !
1 !                 organization independent code for indexed file open
1 !
1 !
1 ! ENVIRONMENT:
1 !
1 !                VAX/VMS OPERATING SYSTEM
1 !
1 !--
1
1 !
1 ! AUTHOR:        Wendy Koenig     CREATION DATE:          24-MAR-78  13:20
1 !
1 !
1 ! MODIFIED BY:
1 !
1 !          V03-016 RAS0284          Ron Schaefer            30-Mar-1984
1 !                  Fix STV value on error paths for RMS$_RPL and RMS$_WPL errors.
1 !
1 !          V03-015 DAS0001          David Solomon           25-Mar-1984
1 !                  Fix broken branch to RM$ALDBUF.
1 !
1 !          V03-014 SHZ0001          Stephen H. Zalewski     27-Feb-1984
1 !                  If you allocate a BDB, you MUST bump the local buffer count
1 !                  (IFB$W_AVLCL).
1 !
1 !          V03-013 JWT0141          Jim Teague              11-Nov-1983
1 !                  Oops, IFB$V_RUM changed to IFB$V_ONLY_RU
1 !
1 !          V03-012 JWT0140          Jim Teague              11-Nov-1983
1 !                  Must check more than one RU bit, as was done in
1 !                  V03-010.
1 !
1 !          V03-011 MCN0013          Maria del C. Nasr       24-Feb-1983
1 !                  Reorganize linkages.
1 !
1 !          V03-010 TMK0005          Todd M. Katz            20-Jan-1983
1 !                  Add support for RMS Journalling and Recovery of ISAM files.
1 !                  For $OPEN this boils down to not allowing a prologue 1 or 2
1 !                  file to be opened if it is marked for any type of journalling.
1 !
1 !          V03-009 KBT0464          Keith B. Thompson       13-Jan-1983
1 !                  Get BKS from key descriptors to aviod LCL bugchecks due
1 !                  to wrong file header info
1 !
1 !          V03-008 KBT0460          Keith B. Thompson       12-Jan-1983
1 !                  Allocate a buffer for reading in prologue (it use to use
1 !                  the buffer allocated for the fwa)
1 !
1 !          V03-007 KBT0225          Keith B. Thompson       23-Aug-1982
1 !                  Reorganize psects
1 !
1 !          V03-006 TMK0004          Todd M. Katz            18-Aug-1982
```

```
1 !
1 !         V03-006 TMK0004          Todd M. Katz              18-Aug-1982
1 !                 Allow prologue 3 files with alternate indicies to be opened.
1 !
1 !         V03-005 TMK0003          Todd M. Katz              01-Jul-1982
1 !                 Implement RMS cluster solution for next record positioning.
1 !                 This emans that RMS no longer has to zero the pointer to the
1 !                 NRP cell in the IFAB, IFB$L_NRP_PTR, because the next record
1 !                 positioning context is now kept locally in the IRAB instead
1 !                 of in a separate systemwide location.
1 !
1 !         V03-004 MCN0012          Maria del C. Nasr         29-Jun-1982
1 !                 Allow different key data types for prologue 3 files.
1 !                 This undoes part of TMK0002.
1 !
1 !         V03-003 KBT0054          Keith B. Thompson         8-Jun-1982
1 !                 Allocate index blocks on all but BIO or UFO opens
1 !
1 !         V03-002 TMK0002          Todd M. Katz              06-May-1982
1 !                 I added code to prevent prologue 3 files with key types
1 !                 other than string and/or alternate indicies from being opened.
1 !                 This code is required for V3A - V3B compatibility, it will go
1 !                 out as a V3.1 patch, and it must be removed for V3B when
1 !                 alternate data types and indicies are supported. The error that
1 !                 will be returned is: error in prologue version.
1 !
1 !                 I also fixed up some of the error paths which were not
1 !                 releasing all accessed VBNs of the file before returning
1 !                 their appropriate error.
1 !
1 !         V03-001 TMK0001          Todd M. Katz              24-Mar-1982
1 !                 Change all references from IFB$B_KBUFSZ to IFB$W_KBUFSZ.
1 !
1 !         V02-020 CDS0005          C D Saether               5-Feb-1982
1 !                 Back out V02-019.  GBC is now a record attribute.
1 !
1 !         V02-019 CDS0004          C D Saether               3-Jan-1982
1 !                 Return GBC field from prologue.
1 !
1 !         V02-018 CDS0003          C D Saether               9-Aug-1981
1 !                 Use alternate linkage declaration for RELEASE.
1 !
1 !         V02-017 CDS0002          C D Saether               16-Jul-1981
1 !                 Remove check for ppf file.
1 !
1 !         V02-016 MCN0011          Maria del C. Nasr         05-Jun-1981
1 !                 Make keybuffer size 2 bytes longer for compressed indexes,
1 !                 and primary key.
1 !
1 !         V02-015 PSK0002          P S Knibbe                20-Apr-1981
1 !                 Change some variable names
1 !
1 !         V02-014 PSK0001          P S Knibbe                17-Mar-1981
1 !                 Change the prologue number check to allow prologue 3
1 !                 Change check_two to make sure that at least two index records
1 !                 can fit into an index bucket.
1 !
```

```
!         V02-013 REFORMAT         R A SCHAEFER              23-Jul-1980      14:09
!                 Reformat the source
!
!         V02-012 CDS0001          C D SAETHER               13-MAR-1980
!                 fix V011 fix to check bio in ifab, not fab
!
!         V02-011 RAS0000          Ron Schaefer              27-NOV-79        09:30
!                 Allow BIO access to any device (i.e. magtape), do not read
!                 prolog if so.
!
!         V02-010 CDS0000          Chris Saether,            26-jun-79  17:55
!                 don't allocate stuff if UFO set
!
!****

LIBRARY 'RMSLIB:RMS';

REQUIRE 'RMSSRC:RMSIDXDEF';

! define default psects for code
!
PSECT
    CODE = RM$RMS3(PSECT_ATTR),
    PLIT = RM$RMS3(PSECT_ATTR);

! define linkages
!
LINKAGE
    L_ALDBUF,
    L_CACHE,
    L_CHKSUM,
    L_FABREG,
    L_LINK_7_10_11,
    L_RELEASE_FAB,
    RL$CHECK_TWO            = JSB (REGISTER = 6) :
                              GLOBAL (R_FAB,R_IFAB);

! forward routine
!

FORWARD ROUTINE
    RM$OPEN3B               : RL$FABREG,
    CHECK_TWO               : RL$CHECK_TWO;

! external routines
!
EXTERNAL ROUTINE
    RM$ALDBUF               : RL$ALDBUF ADDRESSING_MODE( LONG_RELATIVE ),
    RM$CHKSUM               : RL$CHKSUM,
    RM$CACHE                : RL$CACHE,
    RM$CLOSE3               : RL$LINK_7_10_11,
    RM$RELEASE              : RL$RELEASE_FAB,
    RM$AL_KEY_DESC          : RL$LINK_7_TO_11;
```

```
1 %SBTTL  'RM$OPEN3B'
1 GLOBAL ROUTINE RM$OPEN3B : RL$FABREG =
1 !++
1 !
1 ! FUNCTIONAL DESCRIPTION:
1 !
1 !               This routine performs the file open functions that are
1 !               specific to the indexed file organization, including:
1 !
1 !               1 -- reading in the prologue
1 !                         and setting up various fields in the FAB and IFAB
1 !               2 -- setting up the index descriptors
1 !                       (linked off the IFAB) and counting the keys
1 !               3 -- determining the size of the  key buffers
1 !                       and setting kbufsz (IFAB) appropriately
1 !
1 !
1 ! CALLING SEQUENCE:
1 !
1 !       enters via case branch from RMS$OPEN and jsb from RM$OPEN3
1 !       returns via rsb to RM$COPRTN.
1 !
1 ! INPUT PARAMETERS:
1 !       none
1 !
1 ! IMPLICIT INPUTS:
1 !
1 !       R11     IMPURE AREA address
1 !       R9      IFAB address
1 !       R8      FAB address
1 !       the contents of the FAB
1 !
1 ! OUTPUT PARAMETERS:
1 !       none
1 !
1 ! IMPLICIT OUTPUTS:
1 !
1 !       R10 is the address of the IFAB
1 !       various fields in the IFAB and FAB are initialized
1 !       index descriptors are allocated
1 !
1 ! ROUTINE VALUE:
1 !
1 !       standard rms, in particular SUC,PLG,RPL,IFA,KSI,ENV
1 !
1 ! SIDE EFFECTS:
1 !
1 !       May wait quite some time for prologue to become free initially.
1 !       Allocates index descriptors
1 !       In the case of an error, key descriptors are deallocated
1 !       R1 - R5  may be destroyed
1 !
1 !--
1
2   BEGIN
2
2   ! Define common registers
```

```
    2     !
    2     EXTERNAL REGISTER
    2         COMMON_FAB_STR;
    2
    2     GLOBAL REGISTER
    2         COMMON_IO_STR;
    2
    2     IFAB = .IFAB_FILE;
    2
    2     ! Have to zero this since it has a conflicting earlier use in the parse
    2     !
    2     IFAB [ IFB$L_IDX_PTR ] = 0;
    2
    2     ! Allocate a BDB in preparation for reading in the prologue.  Even if we
    2     ! do not use it here, it may be used for XAB processing later on.
    2     !
    2     RETURN_ON_ERROR( RM$ALDBUF( 512 ) );                    ! Get a BDB.
    2     IFAB[IFB$W_AVLCL] = .IFAB[IFB$W_AVLCL] + 1;             ! Bump the local buffer count.
    2
    2     ! if UFO or BIO open then quit right here before descriptors get allocated
    2     !
    2     IF  .FAB [ FAB$V_UFO ] OR .IFAB [ IFB$V_BIO ]
    2     THEN
    2         RETURN RMSSUC( SUC );
    2
    2     ! Read in the prologue 1 block which also has the first key descriptor
    2     !
P   2     RETURN_ON_ERROR( CACHE( 1,512 ),
P   2                         BEGIN
P   2                         IF .FAB [FAB$L_STV] EQL 0
P   2                         THEN
P   2                             FAB [FAB$L_STV] = .STATUS OR 1^16;
P   2                         STATUS = RMSERR (RPL)
    2                         END );
    2
    2     RETURN_ON_ERROR( RM$CHKSUM() );
    2
    2     ! Check for correct prologue version
    2     !
    2     IF .BKT_ADDR [ PLG$W_VER_NO ] GTRU PLG$C_VER_3
    2     THEN
    3         BEGIN
    3         RM$RELEASE(0);
    4         RETURN RMSERR( PLV )
    3         END;
    2
    2     ! Do not allow this file to be opened if it is a prologue 1 or 2 file, and
    2     ! any type of RMS Journalling is enabled.
    2
    2     IF   .BKT_ADDR[PLG$W_VER_NO] LSSU PLG$C_VER_3
    2          AND
    3         (.IFAB[IFB$V_RU]
    3                  OR
    3                 .IFAB[IFB$V_ONLY_RU]
    3                  OR
    3                 .IFAB[IFB$V_AT]
    3                  OR
```

```
   3                       .IFAB[IFB$V_BI]
   3                        OR
   3                       .IFAB[IFB$V_AI])
   2          THEN
   3              BEGIN
   3              RM$RELEASE(0);
   3              RETURN RMSERR(ENV);
   2              END;
   2
   2          ! We now have a good prologue in memory
   2          !
   2          IFAB [ IFB$B_PLG_VER ] = .BKT_ADDR [ PLG$W_VER_NO ];
   2          IFAB [ IFB$B_AVBN ] = .BKT_ADDR [ PLG$B_AVBN ];
   2          IFAB [ IFB$B_AMAX ] = .BKT_ADDR [ PLG$B_AMAX ];
   2          IFAB [ IFB$W_FFB ] = 0;
   2
   2          ! Allocate and count index descriptors, determine size of key buffers
   2          !
   3          BEGIN
   3
   3          GLOBAL REGISTER
   3              R_IDX_DFN;
   3
   3          LOCAL
   3              IDX_COMPR,
   3              KEY_DESC          : REF BBLOCK;
   3
   3          ! Index descriptor for primary key the primary key obviously is the largest
   3          ! to date, so set kbufsz
   3          !
   3          IFAB [ IFB$W_KBUFSZ ] = .BKT_ADDR [ KEY$B_KEYSZ ];
   3
   3          ! Start off finding the largest bucket size for key 0
   3          !
   3          IF .BKT_ADDR [ KEY$B_IDXBKTSZ ] GTRU .BKT_ADDR [ KEY$B_DATBKTSZ ]
   3          THEN
   3              IFAB [ IFB$B_BKS ] = .BKT_ADDR [ KEY$B_IDXBKTSZ ]
   3          ELSE
   3              IFAB [ IFB$B_BKS ] = .BKT_ADDR [ KEY$B_DATBKTSZ ];
   3
   3          ! Assume no compression
   3          !
   3          IDX_COMPR = 0;
   3
   3          ! Allocate the primary key descriptor
   3          !
   3          RETURN_ON_ERROR( RM$AL_KEY_DESC( .BKT_ADDR,1,0 ), RM$RELEASE(0) );
   3
   3          IFAB [ IFB$B_NUM_KEYS ] = 1;
   3
   3          KEY_DESC = .BKT_ADDR;
   3
P  3          RETURN_ON_ERROR( CHECK_TWO(),
P  3                           BEGIN
P  3                           RM$CLOSE3();
P  3                           RM$RELEASE(0)
   3                           END );
```

```
	! If the index or primary key is compressed, set flag.
	!
	IF .KEY_DESC [ KEY$V_IDX_COMPR ] OR .KEY_DESC [ KEY$V_KEY_COMPR ]
	THEN
	    IDX_COMPR = 1;

	! Get index descriptors for all other keys, block by block
	!
	WHILE .KEY_DESC [ KEY$L_IDXFL ] NEQ 0
	DO
	    BEGIN

	    LOCAL
		VBN,
		OFFSET;

	    ! Save the vbn and the offset which is in this block
	    !
	    VBN = .KEY_DESC [ KEY$L_IDXFL ];
	    OFFSET = .KEY_DESC [ KEY$W_NOFF ];

	    ! Release current block and the new one
	    !
	    RETURN_ON_ERROR( RM$RELEASE(0) );

	    RETURN_ON_ERROR( CACHE( .VBN,512 ),
				BEGIN
				IF .FAB [FAB$L_STV] EQL 0
				THEN
				    FAB [FAB$L_STV] = .STATUS OR 1^16;
				STATUS = RMSERR (RPL)
				END );


	    RETURN_ON_ERROR( RM$CHKSUM() );

	    ! Loop for all of the key descriptors in this block
	    !
	    DO
		BEGIN

		! Set the pointer to the new key descriptor
		!
		KEY_DESC = .BKT_ADDR + .OFFSET;

		RETURN_ON_ERROR( CHECK_TWO(),
				    BEGIN
				    RM$CLOSE3();
				    RM$RELEASE(0)
				    END );

		! We have a good one so count it
		!
		IFAB [ IFB$B_NUM_KEYS ] = .IFAB [ IFB$B_NUM_KEYS ] + 1;

		! Set the largest key size
  5                     !
  5                     IF .KEY_DESC [ KEY$B_KEYSZ ] GTRU .IFAB [ IFB$W_KBUFSZ ]
  5                     THEN
  5                         IFAB [ IFB$W_KBUFSZ ] = .KEY_DESC [ KEY$B_KEYSZ ];
  5
  5                     ! Set the largest bucket size
  5                     !
  5                     IF .KEY_DESC [ KEY$B_IDXBKTSZ ] GTRU .IFAB [ IFB$B_BKS ]
  5                     THEN
  5                         IFAB [ IFB$B_BKS ] = .KEY_DESC [ KEY$B_IDXBKTSZ ];
  5
  5                     IF .KEY_DESC [ KEY$B_DATBKTSZ ] GTRU .IFAB [ IFB$B_BKS ]
  5                     THEN
  5                         IFAB [ IFB$B_BKS ] = .KEY_DESC [ KEY$B_DATBKTSZ ];
  5
  5                     ! This index descriptor is ok so allocate one in memory
  5                     !
P 5                     RETURN_ON_ERROR( RM$AL_KEY_DESC( .KEY_DESC,.VBN,.OFFSET ),
  5                                      RM$RELEASE(0) );
  5
  5                     ! If there is compression on note it
  5                     !
  5                     IF .KEY_DESC [ KEY$V_IDX_COMPR ]
  5                     THEN
  5                         IDX_COMPR = 1;
  5
  5                     ! Get the offset to the next key descriptor
  5                     !
  5                     OFFSET = .KEY_DESC [ KEY$W_NOFF ]
  5
  5                     END
  5
  5                 ! Leave the loop if the next key descriptor is in another block
  5                 !
  4                 UNTIL .KEY_DESC [ KEY$L_IDXFL ] NEQ .VBN
  4
  3                 END;
  3
  3             ! If any of the keys have the index compressed, then increase the buffer
  3             ! size by two bytes, to store the length and compression counts.
  3             !
  3             IF .IDX_COMPR
  3             THEN
  3                 IFAB [ IFB$W_KBUFSZ ] = .IFAB [ IFB$W_KBUFSZ ] + 2
  3
  2             END;
  2
  2             RETURN_ON_ERROR( RM$RELEASE(0) );
  2
  3             RETURN RMSSUC()
  3
  1             END;


                                                        .TITLE  RM3OPEN
                                                        .IDENT  \V04-000\
                                                        .EXTRN  RM$ALDBUF, RM$CHKSUM
                                                        .EXTRN  RM$CACHE, RM$CLOSE3
                                                        .EXTRN  RM$RELEASE, RM$AL_KEY_DESC

                                                        .PSECT  RM$RMS3,NOWRT,  GBL,  PIC,2

                           00FC   8F   BB 00000 RM$OPEN3B::
                                                        PUSHR   #^M<R2,R3,R4,R5,R6,R7>                          ; 0261
                  5E              14   C2 00004         SUBL2   #20, SP                                         ;
                  5A              59   D0 00007         MOVL    IFAB_FILE, IFAB                                 ; 0324
                           00AC   CA   D4 0000A         CLRL    172(IFAB)                                       ; 0328
                  55       0200   8F   3C 0000E         MOVZWL  #512, R5                                        ; 0333
                      00000000G   EF   16 00013         JSB     RM$ALDBUF                                       ;
                  68              50   E9 00019         BLBC    STATUS, 7$                                      ;
                           0084   CA   B6 0001C         INCW    132(IFAB)                                       ; 0334
 03       06      A8              01   E1 00020         BBC     #1, 6(FAB), 2$                                  ; 0338
                           01A7   31 00025 1$:          BRW     35$                                             ;
 F8       22      AA              05   E0 00028 2$:     BBS     #5, 34(IFAB), 1$                                ;
                                  53   D4 0002D         CLRL    R3                                              ; 0350
                  52       0200   8F   3C 0002F         MOVZWL  #512, R2                                        ;
                  51              01   D0 00034         MOVL    #1, R1                                          ;
                          0000G   30 00037              BSBW    RM$CACHE                                        ;
                  0B              50   E8 0003A         BLBS    STATUS, 4$                                      ;
                             0C   A8   D5 0003D         TSTL    12(FAB)                                         ;
                                  03   13 00040         BEQL    3$                                              ;
                           00F2   31 00042              BRW     20$                                             ;
                           00E6   31 00045 3$:          BRW     19$                                             ;
                          0000G   30 00048 4$:          BSBW    RM$CHKSUM                                       ; 0352
                  7C              50   E9 0004B         BLBC    STATUS, 11$                                     ;
                  03         74   A5   B1 0004E         CMPW    116(BKT_ADDR), #3                               ; 0356
                                  0C   1B 00052         BLEQU   5$                                              ;
                                  53   D4 00054         CLRL    R3                                              ; 0359
                          0000G   30 00056              BSBW    RM$RELEASE                                      ;
                  50       872C   8F   3C 00059         MOVZWL  #34604, R0                                      ; 0360
                                  6A   11 0005E         BRB     11$                                             ;
                                  24   1E 00060 5$:     BGEQU   8$                                              ; 0366
                  50       00A0   CA   9E 00062         MOVAB   160(IFAB), R0                                   ; 0368
 0F               60              01   E0 00067         BBS     #1, (R0), 6$                                    ;
                  0C              60   E8 0006B         BLBS    (R0), 6$                                        ; 0370
 08               60              04   E0 0006E         BBS     #4, (R0), 6$                                    ; 0372
 04               60              02   E0 00072         BBS     #2, (R0), 6$                                    ; 0374
 0C               60              03   E1 00076         BBC     #3, (R0), 8$                                    ; 0376
                                  53   D4 0007A 6$:     CLRL    R3                                              ; 0379
                          0000G   30 0007C              BSBW    RM$RELEASE                                      ;
                  50       8724   8F   3C 0007F         MOVZWL  #34596, R0                                      ; 0380
                                  65   11 00084 7$:     BRB     13$                                             ;
         00B7     CA         74   A5   90 00086 8$:     MOVB    116(BKT_ADDR), 183(IFAB)                        ; 0385
         00B0     CA         66   A5   B0 0008C         MOVW    102(BKT_ADDR), 176(IFAB)                        ; 0386
                             5C   AA   B4 00092         CLRW    92(IFAB)                                        ; 0388
         00B4     CA         14   A5   9B 00095         MOVZBW  20(BKT_ADDR), 180(IFAB)                         ; 0404
           0B     A5         0A   A5   91 0009B         CMPB    10(BKT_ADDR), 11(BKT_ADDR)                      ; 0408
                                  07   1B 000A0         BLEQU   9$                                              ;
           5E     AA         0A   A5   90 000A2         MOVB    10(BKT_ADDR), 94(IFAB)                          ; 0410
                                  05   11 000A7         BRB     10$                                             ;
           5E     AA         0B   A5   90 000A9 9$:     MOVB    11(BKT_ADDR), 94(IFAB)                          ; 0412
                             10   AE   D4 000AE 10$:    CLRL    IDX_COMPR                                       ; 0416
                  7E              01   7D 000B1         MOVQ    #1, -(SP)                                       ; 0420
```

```
                                55 DD 000B4        PUSHL   BKT_ADDR                      ;
                             0000G 30 000B6        BSBW    RM$AL_KEY_DESC                ;
                 5E             0C C0 000B9        ADDL2   #12, SP                       ;
                 56             50 D0 000BC        MOVL    R0, STATUS                    ;
                 0A             56 E8 000BF        BLBS    STATUS, 12$                   ;
                                53 D4 000C2        CLRL    R3                            ;
                             0000G 30 000C4        BSBW    RM$RELEASE                    ;
                 50             56 D0 000C7        MOVL    STATUS, R0                    ;
                                70 11 000CA 11$:   BRB     21$                           ;
            00B2 CA             01 90 000CC 12$:   MOVB    #1, 178(IFAB)                 ; 0422
                 56             55 D0 000D1        MOVL    BKT_ADDR, KEY_DESC            ; 0424
                             0000V 30 000D4        BSBW    CHECK_TWO                     ; 0430
              0C AE             50 D0 000D7        MOVL    R0, STATUS                    ;
                 0E       0C    AE E8 000DB        BLBS    STATUS, 14$                   ;
                             0000G 30 000DF        BSBW    RM$CLOSE3                     ;
                                53 D4 000E2        CLRL    R3                            ;
                             0000G 30 000E4        BSBW    RM$RELEASE                    ;
                 50       0C    AE D0 000E7        MOVL    STATUS, R0                    ;
                                4F 11 000EB 13$:   BRB     21$                           ;
      05      10 A6             03 E0 000ED 14$:   BBS     #3, 16(KEY_DESC), 15$         ; 0434
      04      10 A6             06 E1 000F2        BBC     #6, 16(KEY_DESC), 16$         ;
              10 AE             01 D0 000F7 15$:   MOVL    #1, IDX_COMPR                 ; 0436
              04 AE       04    A6 9E 000FB 16$:   MOVAB   4(R6), 4(SP)                  ; 0451
                                66 D5 00100 17$:   TSTL    (KEY_DESC)                    ; 0440
                                03 12 00102        BNEQ    18$                           ;
                              00B7 31 00104        BRW     33$                           ;
              0C AE             66 D0 00107 18$:   MOVL    (KEY_DESC), VBN               ; 0450
              08 AE       04    BE 3C 0010B        MOVZWL  @4(SP), OFFSET                ; 0451
                                53 D4 00110        CLRL    R3                            ; 0455
                             0000G 30 00112        BSBW    RM$RELEASE                    ;
                 29             50 E9 00115        BLBC    STATUS, 23$                   ;
                                53 D4 00118        CLRL    R3                            ; 0463
                 52     0200    8F 3C 0011A        MOVZWL  #512, R2                      ;
                 51       0C    AE D0 0011F        MOVL    VBN, R1                       ;
                             0000G 30 00123        BSBW.   RM$CACHE                      ;
                 15             50 E8 00126        BLBS    STATUS, 22$                   ;
                          0C    A8 D5 00129        TSTL    12(FAB)                       ;
                                09 12 0012C        BNEQ    20$                           ;
0C    A8         50 00010000    8F C9 0012E 19$:   BISL3   #65536, STATUS, 12(FAB)       ;
                 50     C104    8F 3C 00137 20$:   MOVZWL  #49412, STATUS                ;
                                62 11 0013C 21$:   BRB     30$                           ;
                             0000G 30 0013E 22$:   BSBW    RM$CHKSUM                     ; 0466
                 5C             50 E9 00141 23$:   BLBC    STATUS, 30$                   ;
      56         55       08    AE C1 00144 24$:   ADDL3   OFFSET, BKT_ADDR, KEY_DESC    ; 0475
                             0000V 30 00149        BSBW    CHECK_TWO                     ; 0481
                 6E             50 D0 0014C        MOVL    R0, STATUS                    ;
                 05             6E E8 0014F        BLBS    STATUS, 25$                   ;
                             0000G 30 00152        BSBW    RM$CLOSE3                     ;
                                41 11 00155        BRB     29$                           ;
                        00B2    CA 96 00157 25$:   INCB    178(IFAB)                     ; 0485
                 50       14    A6 9A 0015B        MOVZBL  20(KEY_DESC), R0              ; 0489
            00B4 CA             50 B1 0015F        CMPW    R0, 180(IFAB)                 ;
                                06 1B 00164        BLEQU   26$                           ;
            00B4 CA       14    A6 9B 00166        MOVZBW  20(KEY_DESC), 180(IFAB)       ; 0491
              5E AA       0A    A6 91 0016C 26$:   CMPB    10(KEY_DESC), 94(IFAB)        ; 0495
                                05 1B 00171        BLEQU   27$                           ;
              5E AA       0A    A6 90 00173        MOVB    10(KEY_DESC), 94(IFAB)        ; 0497
```

```
                                                     B 10
RM3OPEN                                           16-Sep-1984 01:54:23    VAX-11 Bliss-32 V4.0-742                  Page  12
V04-000         RM$OPEN3B                         14-Sep-1984 13:01:32    DISK$VMSMASTER:[RMS.SRC]RM3OPEN.B32;1      (3)

                  5E   AA       0B   A6  91 00178 27$:      CMPB    11(KEY_DESC), 94(IFAB)                    ; 0499
                                     05  1B 0017D           BLEQU   28$                                       ;
                  5E   AA       0B   A6  90 0017F           MOVB    11(KEY_DESC), 94(IFAB)                    ; 0501
                                08   AE  DD 00184 28$:      PUSHL   OFFSET                                    ; 0506
                                10   AE  DD 00187           PUSHL   VBN                                       ;
                                     56  DD 0018A           PUSHL   KEY_DESC                                  ;
                                  0000G  30 0018C           BSBW    RM$AL_KEY_DESC                            ;
                       5E            0C  C0 0018F           ADDL2   #12, SP                                   ;
                       6E            50  D0 00192           MOVL    R0, STATUS                                ;
                       0A            6E  E8 00195           BLBS    STATUS, 31$                               ;
                                     53  D4 00198 29$:      CLRL    R3                                        ;
                                  0000G  30 0019A           BSBW    RM$RELEASE                                ;
                       50            6E  D0 0019D           MOVL    STATUS, R0                                ;
                                     30  11 001A0 30$:      BRB     36$                                       ;
            04    10   A6            03  E1 001A2 31$:      BBC     #3, 16(KEY_DESC), 32$                     ; 0510
                  10   AE            01  D0 001A7           MOVL    #1, IDX_COMPR                             ; 0512
                  04   AE       04   A6  9E 001AB 32$:      MOVAB   4(R6), 4(SP)                              ; 0516
                  08   AE       04   BE  3C 001B0           MOVZWL  @4(SP), OFFSET                            ;
                  0C   AE            66  D1 001B5           CMPL    (KEY_DESC), VBN                           ; 0522
                                     89  13 001B9           BEQL    24$                                       ;
                                   FF42  31 001BB           BRW     17$                                       ; 0470
                       05       10   AE  E9 001BE 33$:      BLBC    IDX_COMPR, 34$                            ; 0529
                0084   CA            02  A0 001C2           ADDW2   #2, 180(IFAB)                             ; 0531
                                     53  D4 001C7 34$:      CLRL    R3                                        ; 0535
                                  0000G  30 001C9           BSBW    RM$RELEASE                                ;
                       03            50  E9 001CC           BLBC    STATUS, 36$                               ;
                       50            01  D0 001CF 35$:      MOVL    #1, R0                                    ; 0537
                       5E            14  C0 001D2 36$:      ADDL2   #20, SP                                   ; 0539
                              00FC   8F  BA 001D5           POPR    #^M<R2,R3,R4,R5,R6,R7>                    ;
                                     05  001D9              RSB                                               ;

; Routine Size:  474 bytes,    Routine Base:  RM$RMS3 + 0000


;  478          0540  1
```

```
1 %SBTTL  'CHECK_TWO'
1 ROUTINE CHECK_TWO ( KEY_DESC : REF BBLOCK ) : RL$CHECK_TWO =
1 !++
1 !
1 ! FUNCTIONAL DESCRIPTION:
1 !
1 !       Check to make sure that at least two records will fit in
1 !       each index, if not don't even let the user open the file
1 !       since it will only lead to trouble later
1 !       note: create does check this but rms-11 doesn't
1 !       if we release w/ a new rms-11 that does there would be no way of
1 !       creating such files and we could take the check out
1 !
1 ! CALLING SEQUENCE:
1 !
1 !       CHECK_TWO( KEY_DESC )
1 !
1 ! INPUT PARAMETERS:
1 !
1 !       KEY_DESC -- pointer to the on-disk key descriptor
1 !
1 ! IMPLICIT INPUTS:
1 !
1 !       FAB -- so that in case of an error, the guilty key of reference
1 !               can be passed back in the stv
1 !
1 ! OUTPUT PARAMETERS:
1 !       none
1 !
1 ! IMPLICIT OUTPUTS:
1 !       none
1 !
1 ! ROUTINE VALUE:
1 !
1 !       KSI if two keys will not fit in the index
1 !       rmssuc if they will
1 !
1 ! SIDE EFFECTS:
1 !       none
1 !
1 !--
1
2     BEGIN
2
2     EXTERNAL REGISTER
2         R_IFAB_STR,
2         R_FAB_STR;
2
2     ! Make sure at least 2 keys will fit in the index level
2     !
2     LOCAL
2         KEYSZ,              ! Size of key
2         BYTES;              ! Number of bytes available in bucket
2
2     BYTES = ( .KEY_DESC [ KEY$B_IDXBKTSZ ] * 512 ) - BKT$C_OVERHDSZ - 1;
2     KEYSZ = .KEY_DESC [ KEY$B_KEYSZ ];
2
```

```
;  537      0598  2      IF .IFAB [ IFB$B_PLG_VER ] LSSU PLG$C_VER_3
;  538      0599  2      THEN
;  539      0600  3          BEGIN
;  540      0601  3          IF 2 * ( .KEYSZ + 2 + IRC$C_IDXPTRBAS + IRC$C_IDXOVHDSZ) GTRU .BYTES
;  541      0602  3          THEN
;  542      0603  4              BEGIN
;  543      0604  4              FAB [ FAB$L_STV ] = .KEY_DESC [ KEY$B_KEYREF ];
;  544      0605  4              RETURN RMSERR(KSI);
;  545      0606  3              END;
;  546      0607  3          END
;  547      0608  2      ELSE
;  548      0609  3          BEGIN
;  549      0610  3
;  550      0611  3          BYTES = .BYTES - 3;
;  551      0612  3
;  552      0613  3          IF .KEYSZ LEQU KEY$C_MAX_INDEX
;  553      0614  3          THEN
;  554      0615  4              BEGIN  ! fixed index record
;  555      0616  4
;  556      0617  4              IF 2 * ( .KEYSZ + 4 ) GTRU .BYTES
;  557      0618  4              THEN
;  558      0619  5                  BEGIN
;  559      0620  5                  FAB [ FAB$L_STV ] = .KEY_DESC [ KEY$B_KEYREF ];
;  560      0621  5                  RETURN RMSERR(KSI);
;  561      0622  4                  END;
;  562      0623  4              END
;  563      0624  3          ELSE
;  564      0625  4              BEGIN  ! variable index records
;  565      0626  4
;  566      0627  4              IF 2 * ( .KEYSZ + 4 + 2 ) GTRU .BYTES
;  567      0628  4              THEN
;  568      0629  5                  BEGIN
;  569      0630  5                  FAB [ FAB$L_STV]  = .KEY_DESC [ KEY$B_KEYREF ];
;  570      0631  5                  RETURN RMSERR(KSI);
;  571      0632  4                  END;
;  572      0633  3              END;
;  573      0634  2          END;
;  574      0635  2
;  575      0636  3      RETURN RMSSUC()
;  576      0637  3
;  577      0638  1      END;



                                 0C   BB 00000 CHECK_TWO:
                                                          PUSHR    #^M<R2,R3>                                ; 0542
                      50      0A  A6  9A 00002            MOVZBL   10(KEY_DESC), R0                          ; 0595
         50           50          09  78 00006            ASHL     #9, R0, R0                                ;
                      53      F1  A0  9E 0000A            MOVAB    -15(R0), BYTES                            ;
                      50      14  A6  9A 0000E            MOVZBL   20(KEY_DESC), KEYSZ                       ; 0596
         52           50          01  78 00012            ASHL     #1, KEYSZ, R2                             ; 0601
                      03    00B7  CA  91 00016            CMPB     183(IFAB), #3                             ; 0598
                                  09  1E 0001B            BGEQU    1$                                        ;
                      51      0A  A2  9E 0001D            MOVAB    10(R2), R1                                ; 0601
                      53          51  D1 00021            CMPL     R1, BYTES                                 ;
```

```
                                    15  11 00024        BRB     4$                                        ;
                          53        03  C2 00026 1$:    SUBL2   #3, BYTES                                 ; 0611
                          06        50  D1 00029        CMPL    KEYSZ, #6                                 ; 0613
                                    06  1A 0002C        BGTRU   2$                                        ;
                          50    08  A2  9E 0002E        MOVAB   8(R2), R0                                 ; 0617
                                    04  11 00032        BRB     3$                                        ;
                          50    0C  A2  9E 00034 2$:    MOVAB   12(R2), R0                                ; 0627
                          53        50  D1 00038 3$:    CMPL    R0, BYTES                                 ;
                                    0C  1B 0003B 4$:    BLEQU   5$                                        ;
                     0C   A8    15  A6  9A 0003D        MOVZBL  21(KEY_DESC), 12(FAB)                     ; 0630
                          50  8784  8F  3C 00042        MOVZWL  #34692, R0                                ; 0631
                                    03  11 00047        BRB     6$                                        ;
                          50        01  D0 00049 5$:    MOVL    #1, R0                                    ; 0636
                                    0C  BA 0004C 6$:    POPR    #^M<R2,R3>                                ; 0638
                                        05 0004E        RSB                                               ;

; Routine Size:  79 bytes,    Routine Base:  RM$RMS3 + 01DA


;  578          0639  1
;  579          0640  1 END
;  580          0641  1
;  581          0642  0 ELUDOM




;                                   PSECT SUMMARY
;
;       Name                        Bytes                         Attributes
;
;   RM$RMS3                               553  NOVEC,NOWRT,  RD ,  EXE,NOSHR,  GBL,  REL,  CON,  PIC,ALIGN(2)




;                            Library Statistics
;
;                                         -------- Symbols --------         Pages        Processing
;       File                              Total    Loaded   Percent         Mapped       Time
;
;   _$255$DUA28:[RMS.OBJ]RMS.L32;1         3109       67        2            154          00:00.4




;                                   COMMAND QUALIFIERS

;       BLISS/CHECK=(FIELD,INITIAL,OPTIMIZE)/LIS=LIS$:RM3OPEN/OBJ=OBJ$:RM3OPEN MSRC$:RM3OPEN/UPDATE=(ENH$:RM3OPEN)

; Size:          553 code + 0 data bytes
; Run Time:         00:14.8
; Elapsed Time:     00:39.5
; Lines/CPU Min:     2607
; Lexemes/CPU-Min: 17788
; Memory Used:  193 pages
; Compilation Complete
```

RM3
V04

0326 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

RM3NEXTRE
LIS

RM3OPEN
LIS

RM3POSRFA
LIS

RM3PCKUNP
LIS

RM3POSKEY
LIS

RM3POSSEQ
LIS